UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

| APPLICATION NO. | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|---|---|---|---|---|
| 09/827,971 | 04/06/2001 | Jason Souloglou | | 5417 |

36183 7590 06/06/2005

PAUL, HASTINGS, JANOFSKY & WALKER LLP
P.O. BOX 919092
SAN DIEGO, CA 92191-9092

| EXAMINER | |
|---|---|
| CHOW, CHIH CHING | |
| **ART UNIT** | **PAPER NUMBER** |
| 2192 | |

DATE MAILED: 06/06/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

| Office Action Summary | Application No. | Applicant(s) | |
|---|---|---|---|
| | 09/827,971 | SOULOGLOU ET AL. | |
| | Examiner | Art Unit | |
| | Chih-Ching Chow | 2192 | |

*-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --*

**Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

**Status**

1)☒ Responsive to communication(s) filed on *14 February 2005*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

**Disposition of Claims**

4)☒ Claim(s) *1-18* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-18* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

**Application Papers**

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *14 February 2005* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

**Priority under 35 U.S.C. § 119**

12)☒ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☒ All b)☐ Some * c)☐ None of:

1.☒ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

**Attachment(s)**

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date *03/17/03*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.



AD

## DETAILED ACTION

1. This action is responsive to amendment dated February 14, 2005.

2. Per Applicants' request, FIGs. 5-7 and Specification have been amended.

3. Claims 1-18 remain pending.

### *Response to Amendment*

4. Applicants' Foreign Priority document filed on 02/14/2005, responding to the 08/11/2004 Office action provided in the objection of 'Priority'. The examiner has reviewed the filed Foreign Priority document respectfully.

5. The objection to the priority is hereby withdrawn in view of Applicants' Foreign Priority document, thus the effective priority date for the current application is October 10, 1998, filing date of GB Patent Application No. 9822075.9. The new effective filing date has brought to a new ground of rejection (see below).

6. Applicants' amendment dated 02/14/2005, responding to the 08/11/2004 Office action provided in the objection of drawings. The examiner has reviewed the updated drawings, Figures 5-7, respectfully.

7. The set of formal drawings filed concurrently with the above-mentioned amendment is accepted by the Examiner.

8. The Double Patenting rejections to the copending application S.N. 10/164,772 and 10/165,378 are withdrawn, since preliminary amendments canceling Claims 1-18 for both of S.N. 10/164,772 and 10/165,378 were submitted.

9. Applicants' amendment dated 02/14/2005, responding to the 08/11/2004 Office action provided in the objection of specification. The examiner has reviewed the updated specification respectfully.

10. The objection to the specification is hereby withdrawn in view of Applicants' amendment to the specification.

***Response to Arguments***

11. Applicants' arguments for Claims 1-3, 5-7, 12, 14, 16-18 have been fully considered respectfully by the examiner, the Applicants' arguments are listed as following: a prior art for the 'abstract register' is thus recited as below.

- Claims 1-3, 5-7, 12, 14, and 17, "Aho is referring to physical register implemented in the CPU that runs the code that Aho is compiling. Aho's descrition of allocating and assigning registers concerns physical registers, not register objects representing abstract registers in the intermediate representation." -- a prior art by Koizumi teaches 'abstract register' for intermediate representation is recited as below.
- Claim 5, "Aho's t1, t2, t3, t4 of Fig. 9.18 do not feed into any register objectes", however, see Aho's Fig. 9.19, a , b, c, d, e, t1, all feed into registers R0, R1.
- Claim 7, "Aho fails to teach crating an expression only once and referencing the expression to all register objects to which it relates." - See citation of Koizumi in Claim 7 rejection.

- Claims 16 and 18, "Aho fails to teach or suggest actually generating register objects in an intermediate representation for holding variable values" -- see citation of Koizumi in Claim 16 rejection.
- Lethin's filing date is 10/21/1998, which is after the effective filing date of the current application, which is 10/10/1998. Koizumi is recited below instead of Lethin, since the certified copy of GB Patent Application No. 9822075.9 was not submitted originally.

12. Examiner is maintaining the 35 USC § 103 Rejections. For the Applicants' convenience they are listed as following,.

***Claim Rejections - 35 USC § 103***

13. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

14. Claims 1-18 are rejected under 35 U.S.C. 103(a) as being unpatentable over Aho et al, "Compiler, principles, techniques, and tools" book, published in 1986 (herein after "Aho"), in view of of U.S. Patent No. 5,586,323 by Shinobu Koizumi et al. (hereinafter "Koizumi").

**CLAIMS**

1. A method of generating an intermediate representation of program code, the method comprising the computer implemented steps of:

a. generating a plurality register objects representing abstract registers, a single register object representing a respective abstract register; and

b. generating expression objects each representing a different element of said program code as that element arises in the program code, each expression object being referenced by a register object which it relates either directly, indirectly via references from other expression objects.

**Aho / Koizumi**

With respect to Claim 1, first paragraph, Aho discloses the concept of generating **intermediate representation** of program code in his book, on page 12, section 'Intermediate Code Generation': "After syntax and semantic analysis, some computers generate an explicit **intermediate representation** of the **source program**. We can think of this **intermediate representation** as a program for an abstract machine." With respect to paragraphs a and b.

a. Registers are inherited from any microprocessors, used to hold data for a particular purpose. In Aho's book, Aho further discloses the use of registers for any element of program code. In order for Aho to use those registers, those registers must be **generated** at some point. On page 517, section 'Register Allocation': "efficient utilization of registers is particularly important in generating good code. The use of registers is often subdivided into two subprograms:

1. During register allocation, we select the set of variables that will reside in registers at the point in the program.
2. During a subsequent register assignment phase, we pick the specific register that a variable will reside in. "

Aho's variables are held by **register objects**, they also imply to 'abstract registers' since they can hold any type of data (abstract data types) ;

b. an **expression object** is an operation performed for the register objects. The **expression objects** should be generated when the parsing of the program code. In Aho's book, page 49, under 'Abstract and Concrete Syntax' section, "A useful starting point for thinking about the translation of an input string is an *abstract syntax tree* in which each node represents an operator and the children of the node represent the operands." Here the node can be considered as a **register object** representing a respective **abstract register** and the operator is an **expression object** that is referenced by a **register object** (operand). An example is given in Aho's book, page 558-559, each of the t2, t3, t1 and t4 are 'expression objects' and they are either relates directly, or indirectly via references from other expression objects. (E.g. t1-4 are all **expression objects**, where t2 is indirectly related to t4). Aho teaches all aspects of the applicant's claims but it does not specifically mention 'abstract register'. However, Kiozumi teaches 'abstract register' in an analogous prior art. In Koizumi column 4, lines 53-58, "an **abstract register** machine (also referred to as ARM or Arm in abbreviation) having a plurality of registers is presumed, wherein an instruction sequence for the **abstract register** machine or ARM is made use of as a basic part of the common object

| | |
|---|---|
| | program (referred to as the abstract object program)".<br>It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement the **intermediate representation** of Aho with the abstract register taught by Kiozumi, for the purpose of preserving a form of a program to be executed repeatedly (See Koizumi, column 2, lines 66-67). |
| 2. A method according to claim 1 wherein said program code is expressed in terms of an instruction set of a subject processor. | For the feature of claims 1 see Aho. Aho's book, page 14, "we consider an intermediate form called 'three-address code,' which is like the assembly language for a machine in which every memory location can act like a register. Three-address code consists of **a sequence of instructions**, each of which has at most three operands." This paragraph implies that the original program code is expressed in terms of an **instruction set** of a **subject processor** and a translation will be done for a target processor. |
| 3. A method according to claim 2, wherein said register objects represent abstract registers corresponding to registers of said subject processor. | For the feature of claim 1 and 2 see claim 1 and 2 rejections; as to the "abstract registers corresponding to registers of said subject processor" part, see Koizumi, column 6, lines 25-39, "In response to an **abstract register allocation command for the abstract registers** in the abstract object program, the installer attempts to |

**establish correspondence** between an **abstract register** and a **real register**. In that case, when there exists a real **register for which correspondence with other abstract register** has not been established within a range described in register usage indication of the target machine specifications, i.e. where there is found an idle real register, correspondence is established between the aforementioned **abstract register** and the idle **real register**." Further, " In response to a register releasing or freeing command contained in the abstract object program, the installer clears the **correspondence relation between the abstract register and the real register** (i.e. deallocation is executed by the installer)"

4. A method according to claim 1, wherein each of said steps are performed sequentially for basic blocks of said program code having only one effective entry point instruction and one effective exit point instruction.

For the feature of claim 1 see claim 1 rejection. In Aho, page 528, under 'Basic Blocks' section, "A *basic block* is a **sequence of consecutive statements** in which flow of control enters at the beginning and leaves at the end without halt or possibility of branching except at the end." Aho teaches all aspects of the applicant's claims but it does not specifically mention that the basic blocks of said program code having only one effective entry point instruction and one effective exit point instruction. However, Koizumi teaches a basic block has only one effective entry point instruction and one effective exit point

instruction in an analogous art for the purpose of sending blocks of the original code to the compiler. In Koizumi, column 21, lines 59-62, "At first, an ArmCode instruction sequence of the ArmCode program is divided into a plurality of **basic blocks** by **punctuating the sequence at flow-in points and branching points** (*entry and exit point*) of the control".

5. A method according to claim 1, wherein at least some said expression objects feed into more than one said register object.

For the feature of claim 1 see claim 1 rejection. Again, see Aho, page 559, Fig. 9.20, each of the t2, t3, t1 and t4 are 'expression objects' and they are feed into more than one register object. (E.g. t1 is an expression object, it feed into register objects a and b; t2, is also an expression object, it feed into register objects c and d). Any program would have some expression objects since the program should perform certain functions, functions are 'operations' and they are represented by 'expression objects'; operands feed into operations, here operands are represented by 'register objects'.

6. A method according to claim 1, wherein said expression objects are not duplicated.

For the feature of claim 1 see claim 1 rejection. In Aho page 290, under 'Directed Acyclic Graphs for Expressions' section, "A directed acyclic graph (hereafter called a dag) for an expression identifies the common subexpressions in the expression. Like a syntax tree, a dag has a node for every

| | |
|---|---|
| | subexpression of the expression; an interior node represents an operator and its children represent its operands." In addition, on page 291, first paragraph, "A dag is obtained if the function constructing a node first **checks to see whether an identical node already exists**. ... if so, mknode can return a pointer to the previously constructed node" - the duplication of a function/operation node (expression object) is checked before a new function node is created. |
| 7. A method according to claim 1, wherein a single said expression object is generated for a given element of said program code, and each said expression object is referenced by said register objects to which relates. | For the feature of claim 1 see claim 1 rejection, for the rest of claim 7 feature see Koizumi, column 22, lines 30-64, about abstract register assignment (eliminating duplications), specifically, lines 50-51, "in the instruction 4098, the abstract registers Ar6 and Ar11 are **reused**, while in the instruction 4076, the register Ar7 is reused with the register Ar5 being reused in the instructions 4108, 4110 and 4112, as shown in FIGS. 20 and 21", -- the abstract registers are reused for different expression objects after the optimization processing. |
| 8. A method according to claim 1, wherein if a said register object or a said expression object becomes redundant or unnecessary it is eliminated. | For the feature of claim 1 see claim 1 rejection. For the rest of claim 8 feature see Koizumi column 22, lines 54-59, "In this manner, these abstract registers are replaced by those having the identical values (*redundant or* |

| | |
|---|---|
| | *unnecessary object*), respectively, whereby the instructions for determining the values of the abstract registers Ar8, Ar12, Ar15, Ar9, Ar13, Ar15 and Ar16 are **deleted**." (*redundant or unnecessary expression object is eliminated*)" |
| 9. A method according to claim 8, wherein a redundant or unnecessary said register object or said expression object is identified by maintaining an ongoing count of references being made to that object as a network of register and expression objects is constructed. | For the feature of claim 8 see claim 8 rejection. For the rest of claim 9 feature see Koizumi column 13, lines 13-15, "function: "alloc" (abstract register name, register type, discriminant variable, instruction number, **preserve count**, priority)" - Koizumi keeps track of an ongoing count of references being made to an object. |
| 10. The method according to claim 9, wherein for each expression object count is maintained of the number of references to that expression object from other expression objects or from register objects, the count associated with particular expression object being adjusted each time a reference to that expression object is made or removed. | See rejection of claim 8. |
| 11. A method according to claim 10, wherein an expression object and all references from that expression object are eliminated when said count for that expression object is zero. | See rejection of claim 8. It's obvious that an object is eliminated if the reference count is zero. |
| 12. The method of claim 1, comprising translating the program code written | For the feature of claim 1 see claim 1 rejection. In Aho, page 463, first |

for execution by a processor of a first type so that the program code may be executed by a processor of a second type, using the generated intermediate representation.

paragraph, "In the analysis-synthesis model of a compiler, the front end translates a source program into an **intermediate representation** form which the back end generates target code. Although a source program can be translated directly into the target language, some benefits of using a machine-independent intermediate form are:
1. Retargeting is facilitated; a compiler for a different machine can be created by attaching a back end of the new machine to an existing front end.
2. A machine-independent code optimizer can be done to the intermediate representation." Aho taught us that the program code from a front end processor can be translated into **intermediate representation** and the **intermediate representation can be optimized**, and then be executed to an back end processor. In addition, at the beginning of the current invention spec, 2nd paragraph, the inventor also mentioned that "**Intermediate representation** is a term widely used in the computer industry to refer to terms of abstract computer language in which a program may be expressed, but which is not specific to, and is not intended to be directly executed on, any particular processor...". Since the concept has been 'widely used' therefore it's not an invention.

| | |
|---|---|
| 13. The method claim 12, wherein said translating step is performed dynamically as the program code is run | For the feature of claim 12 see Claim 12 rejection. In Aho, page 347, under 'Static and Dynamic Checking of Types' section, "Checking done by a compiler is said to be static, while checking done when the target program runs is termed **dynamic**." Aho teaches all aspects of the applicant's claims but it does not specifically mention that the translation step is performed as the program code is run. However, Lethin shows |
| 14. The method of claim 1, comprising optimising the program code by optimizing the generated intermediate representation. | For the feature of claim 1 see claim 1 rejection. In Aho page 463, 3rd paragraph, "A machine-independent **code optimizer** can be applied to the **intermediate representation**." |
| 15. The method of claim 14, wherein said optimizing step is used to optimise the program code written for execution by a processor a first type so that the program code may be executed more efficiently by that processor. | For the feature of claim 14 see rejection of claim 14. See Koizumi, column 22, lines 30-64, about abstract register assignment (eliminating duplications), specifically, lines 50-51, "in the instruction 4098, the abstract registers Ar6 and Ar11 are **reused**, while in the instruction 4076, the register Ar7 is reused with the register Ar5 being reused in the instructions 4108, 4110 and 4112, as shown in FIGS. 20 and 21", -- the abstract registers are reused for different expression objects after the optimization processing. |
| 16. A method for generating an intermediate representation of program code written for running on | For items (i) and (ii), see Koizumi column 14, lines 35-60. Koizumi's disclosure teaches using plurality of register |

programmable machine, said method comprising:

(i) generating plurality of register objects for holding variable values to be generated by the program code; and

(ii) generating plurality of expression objects representing fixed values and/or relationships between said fixed values and said variable values according to said program code;

said objects being organized into a branched tree-like network having all register objects at the lowest basic root or tree-trunk level of the network with no register object feeding into any other register object.

objects for holding variable values, and relationships between said fixed values and said variables (position of a variable name in a symbol table or abstract register number representing an address in the memory), for the tree-like network see claim 1 rejection.

17. A system for generating an intermediate representation of program code, comprising:

means for generating a plurality of register objects representing abstract registers, a single register object representing a respective abstract register; and

means for generating expression objects each representing a different element of said program code as that element arises the program code, each expression
object being referenced by a register object to which it relates either directly, or indirectly via references from other expression objects.

See rejection of claim 1.

18. A system for generating an

See rejections of claim 16.

intermediate representation of program code written for running on a programmable machine, the system comprising:

means for generating a plurality of register objects for holding variable values to be generated by the program code; and

means for generating a plurality of expression objects representing fixed values and/or relationships between said fixed values and said variable values according to said program code;

wherein said objects are organised into a branched tree-like network having register objects the lowest basic root or tree-trunk level of the network with no register object feeding into any other register object.

***Conclusion***

15. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

**Lethin** U.S. Patent No. 6,463,582, discloses a method for optimizing objet code translation system.

***Conclusion***

16. The following summarizes the status of the claims:

35 USC § 103 rejection: Claims 1-18

17. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Chih-Ching Chow whose telephone number is 571-272-3693. The examiner can normally be reached on 7:30am - 4:00pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Dam can be reached on 571-272-3695. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306. Any inquiry of a general nature of relating to the status of this application should be directed to the **TC2100 Group receptionist: 571-272-2100.**

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Chih-Ching Chow

Examiner

Art Unit 2192

May 27, 2005

cc Hoang-Anh Antony Nguyen-Ba

ANTONY NGUYEN-BA
PRIMARY EXAMINER